Making it run again next quarter

Turn a working analysis into one that reruns on a new export without editing, fails loudly when the input changes, and can be handed to someone who has none of your context.

Pierre Robentz Cassion

Lesson 8 of 8

Data Analysis Foundations for M&E

What this lesson covers

- The test
- Split the analysis into stages that hand off files
- Put the settings in one place
- Make the pipeline fail loudly
- Parameterise instead of copying
- What never goes in the repository
- The handover README
- Where to go from here

The test

- Next quarter's export arrives.

Split the analysis into stages that hand off files — Example

```
scripts/
  01-read.py         raw export      -> data/interim/muac-typed.parquet
  02-clean.py        typed           -> data/processed/muac-clean.parquet
                                        output/tables/cleaning-log.csv
  03-indicators.py   clean           -> output/tables/gam-by-commune.csv
```

Split the analysis into stages that hand off files — In Python

```
muac.to_parquet("data/interim/muac-typed.parquet", index=False)
```

Split the analysis into stages that hand off files — In R

```
arrow::write_parquet(muac, "data/interim/muac-typed.parquet")
```

Put the settings in one place — In Python

```python
# config.py
from pathlib import Path

RAW = Path("data/raw/muac-screening-artibonite-2024.v1.csv")
INTERIM = Path("data/interim/muac-typed.parquet")
PROCESSED = Path("data/processed/muac-clean.parquet")
TABLES = Path("output/tables")

SAM_MM = 115
GAM_MM = 125
PLAUSIBLE_MUAC = (80, 220)
AGE_RANGE_MONTHS = (6, 59)
MISSING_CODE = "-99"
```

Put the settings in one place — In R

```r
# config.R
RAW       <- "data/raw/muac-screening-artibonite-2024.v1.csv"
INTERIM   <- "data/interim/muac-typed.parquet"
PROCESSED <- "data/processed/muac-clean.parquet"
TABLES    <- "output/tables"

SAM_MM           <- 115
GAM_MM           <- 125
PLAUSIBLE_MUAC   <- c(80, 220)
AGE_RANGE_MONTHS <- c(6, 59)
MISSING_CODE     <- "-99"
```

Make the pipeline fail loudly — In Python (cont.)

```python
def check_input(df):
    expected = {
        "child_id", "commune", "screening_date", "age_months",
        "sex", "muac_mm", "oedema", "outcome",
    }
    missing = expected - set(df.columns)
    if missing:
        raise ValueError(f"export is missing columns: {sorted(missing)}")

    if df["child_id"].isna().any():
        raise ValueError("rows without an identifier")

    measured = df["muac_mm"].dropna()
    if len(measured) == 0:
        raise ValueError("no MUAC measurements at all -- wrong file?")
```

Make the pipeline fail loudly — In Python (cont.)

```python
share_missing = df["muac_mm"].isna().mean()
if share_missing > 0.20:
    raise ValueError(
        f"{share_missing:.1%} of MUAC missing -- above the 20% tolerance. "
        "Investigate before reporting."
    )
```

Make the pipeline fail loudly — In R (cont.)

```r
check_input <- function(df) {
  expected <- c("child_id", "commune", "screening_date", "age_months",
                "sex", "muac_mm", "oedema", "outcome")
  missing <- setdiff(expected, names(df))
  if (length(missing) > 0) {
    stop("export is missing columns: ", paste(missing, collapse = ", "))
  }

  if (any(is.na(df$child_id))) stop("rows without an identifier")

  share_missing <- mean(is.na(df$muac_mm))
  if (share_missing > 0.20) {
    stop(sprintf(
      "%.1f%% of MUAC missing - above the 20%% tolerance. Investigate before reporting.",
      share_missing * 100
    ))
```

Make the pipeline fail loudly — In R (cont.)

```
  }
}
```

Make the pipeline fail loudly

Write the assertion for the failure you would be embarrassed to miss, not for the failure you think is likely. The likely ones you will notice.

Parameterise instead of copying — In Python (cont.)

```
import sys
from pathlib import Path

def build_report(commune: str | None = None):
    df = pd.read_parquet(PROCESSED)
    if commune:
        df = df[df["commune"] == commune]
        if df.empty:
            raise ValueError(f"no rows for commune {commune!r}")

    table = indicator_table(df, "age_band")
    name = commune.lower().replace(" ", "-") if commune else "all"
    table.to_csv(TABLES / f"gam-{name}.csv", index=False)
    return table
```

Parameterise instead of copying — In Python (cont.)

```
if __name__ == "__main__":
    target = sys.argv[1] if len(sys.argv) > 1 else None
    build_report(target)
```

Parameterise instead of copying — In R

```r
build_report <- function(commune = NULL) {
  df <- arrow::read_parquet(PROCESSED)
  if (!is.null(commune)) {
    df <- filter(df, commune == !!commune)
    if (nrow(df) == 0) stop("no rows for commune ", commune)
  }

  table <- indicator_table(df, "age_band")
  name <- if (is.null(commune)) "all" else tolower(gsub(" ", "-", commune))
  readr::write_csv(table, file.path(TABLES, paste0("gam-", name, ".csv")))
  table
}

args <- commandArgs(trailingOnly = TRUE)
build_report(if (length(args) > 0) args[1] else NULL)
```

What never goes in the repository

- **Never commit raw beneficiary data.** Not identified, not pseudonymised. Git keeps every version forever, and deleting...
- **Never commit credentials.** DHIS2 tokens, KoboToolbox API keys, database passwords. Read them from the environment.
- **Add** `data/raw/` **to** `.gitignore` and document where the real file lives — a shared drive, an encrypted volume — in...

What never goes in the repository — Example

```
data/raw/
data/interim/
data/processed/
.env
*.Rhistory
__pycache__/
.Rproj.user/
```

The handover README — Example (cont.)

```
# MUAC screening analysis, Artibonite

## What this produces
Quarterly GAM and SAM rates by commune, sex and age band, with 95% intervals.
Feeds the CMAM site allocation decision and the quarterly donor report.

## Where the data comes from
CommCare export, project `artibonite-nutrition`, form `Community screening`.
Exported monthly by the M&E assistant to the shared drive at <path>.
Not in this repository.

## How to run it
    uv sync
    uv run python scripts/01-read.py
    uv run python scripts/02-clean.py
    uv run python scripts/03-indicators.py
```

The handover README — Example (cont.)

```
## Decisions you should know about
See output/tables/cleaning-log.csv. The one that matters: rows with missing
age are kept, because 60% of the Gros-Morne week of 10-14 June is missing age
and dropping them shifts the commune ranking.

## Who to ask
Indicator definitions: <name>, M&E Manager.
Data collection issues: <name>, Field Coordinator.
```

Where to go from here

- **Data Cleaning and Validation** goes deeper into the profiling and correction work of lessons 5 and 6, including fuzzy…
- **Indicator Design and the LogFrame** starts where lesson 7's reference sheet ends, and works back to the theory of…
- **Survey Analysis, Sampling and Weighting** covers what this course did not: what changes when the data is a…

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-analysis-foundations/foundations-08-reproducing